Superscalar Performance in a Multithreaded Microprocessor

by

Bernard Karl Gunther, BE (Hons)

Department of Computer Science

Submitted in fulfilment of the requirements
for the degree of
Doctor of Philosophy

UNIVERSITY OF TASMANIA
HOBART

December 1993

Statement of Originality

This thesis does not contain any material which has been accepted for the award of any other higher degree or graduate diploma in any tertiary institution. To the best of my knowledge and belief, this thesis does not contain any copy or paraphrase of material previously published or written by another person, except when due reference is made in the text of the thesis.

Access and Copying

B. Smith
19/10/94

Abstract

Multithreaded processors, having hardware support for the concurrent execution of fine-grained threaded computations, are noted for their latency tolerance and low-cost synchronization. Multithreading is a technique for improving the utilization of processing elements (PEs) in parallel processing systems, thereby reducing cost/performance ratios. With increasing integrated circuit densities it is becoming feasible to integrate several PEs onto a single die, and further diminish the physical dimensions of parallel systems. However, by eliminating the artificial on-chip PE boundaries and sharing expensive resources in a more tightly coupled multithreaded architecture, even greater performance can be achieved from similar hardware.

A multithreaded processor architecture (Concurro) was designed for possible microprocessor implementation with the objective of multiple instruction issues per cycle—sustained superscalar performance—by means of multithreading. This thesis considers the trade-offs necessary for such architectures to achieve high throughput and hardware utilization under scalability and cost constraints. A detailed simulation study was carried out to characterize the architecture and evaluate the impact of implementation decisions. The key to efficiency in Concurro is asynchronous, zero-time context switching among a limited set of contexts, promoting effective use of the storage hierarchy. A 64-bit, register-based, load/store instruction set architecture is augmented with thread manipulation primitives and I-structure synchronization operations. Novel cache architectures and controller algorithms were designed for enhancing latency tolerance in the processor, while maximizing utilization of the most costly resources.

When tested on a variety of numerical and integer workloads, Concurro was able to sustain superscalar instruction issue rates for multithreaded operation, yet showed scalar RISC performance on single-thread code. Even with a simple threading strategy it was frequently possible to extract full utilization from functional units or the instruction cache. The architecture showed size scalability to an order of magnitude while remaining binary

compatible across these configurations. Performance of large configurations was shown to be limited ultimately by the bandwidth available from critical shared resources. With an appropriate memory system Concurro attained supercomputer-level floating point throughput operating out of uncached memory. The hardware requirements for this performance are expected to be comparable with those of VLIW machines with similar datapaths.

Acknowledgments

I thank my supervisor, Arthur Sale, for his continued confidence in me, despite my occasional meanderings, and for teaching me the value of independent and critical enquiry. I am indebted to John Morris for reading my drafts with care and enthusiasm, and returning them with valuable feedback. Andrew Partridge's reviews and insights are also very much appreciated.

I am grateful to Carl Lewis for the initial development of the assembler, which helped in getting the first simulations started. Thanks to John Parry for his efforts with the lcc compiler back-end.

The support of many colleagues from the old GCSB will be long remembered. I thank Ed Kazmierczak, Simon Milton, David Wright, Andrew Partridge, Jo Jordan, Tony Dekker, and Ben Lian for their companionship and advice, and for sharing with me their spirit of discovery. In particular, I wish to thank David Wright for introducing to me some fascinating perspectives of computer science.

I am especially grateful to my mother for her selfless help, patience, and interest. I thank her for instilling in me the courage to pursue my goals.

B.K. Gunther
Hobart, 1993

To my mother,
Eva.

Contents